Dissecting Lottery Ticket Transformers: Structural and Behavioral Study of Sparse Neural Machine Translation

Rajiv Movva[*]
MIT
Cambridge, MA
rmovva@mit.edu

Jason Zhao[*]
MIT
Cambridge, MA
jzhao7@mit.edu

Abstract

Recent work on the lottery ticket hypothesis has produced highly sparse Transformers for NMT while maintaining BLEU. However, it is unclear how such pruning techniques affect a model's learned representations. By probing Transformers with more and more low-magnitude weights pruned away, we find that complex semantic information is first to be degraded. Analysis of internal activations reveals that higher layers diverge most over the course of pruning, gradually becoming less complex than their dense counterparts. Meanwhile, early layers of sparse models begin to perform more encoding. Attention mechanisms remain remarkably consistent as sparsity increases.

1 Introduction

In recent years, Transformers (Vaswani et al., 2017) have defined state-of-the-art performance on a variety of NLP tasks, including machine translation (MT) and language modeling. While large Transformer models can learn uniquely rich representations, they are also highly overparameterized (Michel et al., 2019; Hao et al., 2019). Several studies have therefore attempted to prune Transformers during or after training while retaining as much performance as possible (Ganesh et al., 2020). Some methods have been fairly successful, achieving compression ratios up to $10\times$ depending on the downstream task.

Looking beyond task performance, however, it remains unclear how widely-used pruning methods affect a model's learned representations. For example, a pruned Transformer may translate text at the same BLEU, but does pruning affect the model in ways unaccounted for by this metric?

Motivated by this question, we apply recent analysis techniques to study the representations of increasingly sparse Transformers trained on MT. We

[*]Equal contribution.

perform magnitude pruning in an iterative, lottery-ticket fashion to identify Transformers at competitive sparsities with no drop in task performance (Renda et al., 2020; Yu et al., 2020; Brix et al., 2020). We examine the internal structures of our models as sparsity increases, specifically addressing the following questions:

- Does pruning affect what linguistic knowledge is learned by the model?

- How do individual model components (neurons, layers, attentions) change with pruning?

- How is information distributed across layers in sparse vs. dense models?

- What are the differences in pruning dynamics for the three types of model attention (encoder self, encoder-decoder, decoder self)?

Using iterative magnitude pruning (IMP), we train an En-De Transformer that retains 99.4% of BLEU at 66.4% sparsity. During IMP, we obtain eight Transformer models at varying levels of sparsity, along with the original unpruned model. We probe these models' representations for learned linguistic knowledge on eighteen auxiliary syntactic and semantic tasks (Conneau et al., 2018; Liu et al., 2019). We then perform an unsupervised comparison of the representations and attention distributions between dense and sparse models, adopting metrics posed in Wu et al. (2020). Our key conclusions are as follows:

- Complex semantic information is lost first during pruning, before BLEU decreases.

- Model activations steadily diverge from their unpruned representations, particularly at higher layers.

- Information flow between layers becomes more distributed in sparse Transformers: lower layers perform more encoding.

- The encoder-decoder attention is the richest in representation, whereas the decoder self-attention is the simplest. Still, all attention mechanisms remain functionally consistent across sparsities.

2 Related Work

Much work has attempted to reduce the parameter count of dominant Transformer-based architectures (Ganesh et al., 2020). Several papers prune BERT (Devlin et al., 2018), either via structured removal of layers and attention heads (Fan et al., 2019; Sajjad et al., 2020) or unstructured pruning of individual weights (Chen et al., 2020; Gordon et al., 2020). Structured head pruning has also been applied to NMT (Voita et al., 2019; Michel et al., 2019), in which BLEU is used to quantify effective compression. Recent work from Yu et al. (2020) uses iterative magnitude pruning to identify lottery tickets for NMT, retaining 99% of BLEU at 67% sparsity for Transformer-Big. To our knowledge, they achieve the highest net pruning ratio on translation with no drop in performance.

While most such studies are primarily considered with maximizing sparsity, a subset of them address other questions. Gordon et al. (2020) weight prune BERT and finetune on GLUE tasks to identify how much sparsity each task can accommodate. Prasanna et al. (2020) prune heads while finetuning BERT on GLUE tasks, and identify which heads are masked most often. They use pruning as an analysis technique to identify 'good' or 'bad' BERT subnetworks. Similarly, Michel et al. (2019) and Voita et al. (2019) prune heads to identify which types of attention are most relevant to performance. However, these studies focus only on task performance, leaving other behavioral differences between dense and sparse models unexplored.

Relevant methods of analyzing representations in NLP include probing classifiers, which evaluate model representations on supervised tasks for morphology (Belinkov et al., 2017), syntax (Shi et al., 2016), and/or semantics (Voita et al., 2018). For Transformers, some work has directly examined the attention module (Raganato and Tiedemann, 2018; Voita et al., 2019). These analyses include inference of functional annotations for particular heads (Clark et al., 2019), or assessment of atten-

tion's ability to perform unsupervised syntax tree prediction (Kim et al., 2020). Recent work has also applied high-dimensional similarity analysis methods to compare learned representations within or across models (Saphra and Lopez, 2019; Wu et al., 2020). For instance, Bau et al. (2019) identify recurring neurons across NMT models, interpret their functions, and control their activations. A broader survey of such literature is covered by Belinkov and Glass (2018). We leverage some of these representation analysis methods to study and compare sparse and dense Transformers, which, to our knowledge, previous work has not addressed.

3 Generating Lottery Ticket Subnetworks

3.1 Training transformer-based NMT

Following Vaswani et al. (2017), we train Transformer-Big on WMT16-En-De for 60 epochs and achieve detokenized test BLEU of 27.77 on Newstest14. Note that this score appears lower than Vaswani et al. since we do not use compound-split BLEU, which artificially inflates performance[1].

3.2 Iterative pruning & rewinding protocol

Recent work on the lottery ticket hypothesis has demonstrated the efficacy of iterative magnitude pruning (IMP) with *weight rewinding* (Frankle et al., 2019), where unpruned network weights and the learning rate are rewound to values early in training after every pruning iteration. Yu et al. (2020) apply this method to Transformers, and also show that leaving embedding weights unpruned better retains performance. Renda et al. (2020) propose *learning rate (LR) rewinding*, where the learning rate is rewound to a value earlier in training, but the weights remain unchanged. They found that LR rewinding often performs better for deep NMT models and requires fewer training iterations.

Combining insights, we iteratively prune as follows: after training to completion, we mask the 20% lowest magnitude non-embedding weights, rewind the LR to halfway through training (30 epochs), and retrain to completion before another prune. We also trained an iterative random pruning baseline using the same approach. As a clarifying note, we acknowledge that a "lottery ticket" traditionally refers to the network's pruned structure *and* its weights early in training; however, for more

---

[1]When we compute compound-split BLEU with ensembling, our BLEU is 28.74, compromable with Vaswani et al.

convenient referral to our sparse models, we adopt a broader definition of the term to also describe network substructures identified via LR rewinding.

3.3 Lottery ticket performance

We iteratively pruned until our network's performance dropped significantly (Table A1). After seven IMP steps, our model's non-embedding weights were 79% sparse (sparsity including emb. weights: 66.4%), and test BLEU was over 99% of its unpruned value (27.61 & 27.77 respectively). In the subsequent pruning iteration, performance starts to drop more rapidly (0.4 BLEU with only 3% of total additional weights being pruned), suggesting the start of the "power-law" performance decay observed during IMP (Rosenfeld et al., 2020). Our results align closely with Yu et al. (2020), who also report rapid BLEU drop at $> 67\%$ net sparsity. Because we are primarily interested in sparse models that retain full performance, we stopped pruning at this iteration. For downstream experiments, we keep the seven pruned models which experience a negligible performance drop, as well as an eighth model to hint trends as pruning starts to degrade main task performance. In subsequent analyses, we refer to the model after the kth iteration of IMP as LTHk, with LTH0 referring to the unpruned model.

3.4 Where are weights being pruned?

Examining which model components are most readily pruned may hint at their relative importances. Voita et al. (2019) find that late encoder-decoder heads and early decoder-decoder heads are retained the longest. We complement these findings with our results from unstructured pruning.

We compute sparsities of each weight module as overall Transformer sparsity increases (Figures A1, A2). For both the encoder and decoder, later layers exhibit higher fully connected sparsities, with as much as 25% higher sparsity in decoder layer 6 FC weights compared to decoder layer 1 FC weights. This trend suggests that higher layers' FC modules are most overparameterized.

For encoder and decoder self-attention (we compute sparsity across QKV weights $\mathbf{W_q}, \mathbf{W_k}, \mathbf{W_v}$ and out proj. matrix $\mathbf{W_o}$), layer 1 is pruned significantly more than other layers, particularly for low-sparsity models. Layer 6 is pruned next most. Meanwhile, encoder-decoder attention is pruned least across layers compared to all other modules: when overall model sparsity (excl. embeddings)

reaches 79%, enc-dec sparsity is only 55%. Late enc-dec layers are pruned slightly less than early ones. Finally, across all attention types, $\mathbf{W_v}$ and $\mathbf{W_o}$ are 25% sparser than $\mathbf{W_q}$ and $\mathbf{W_k}$, suggesting that projection steps downstream of computing attention weights are particularly overparameterized.

4 Probing for Linguistic Knowledge

4.1 Task setup

Probing classifiers are used to measure latent linguistic knowledge in word embeddings (Belinkov et al., 2019). We extract representations from our Transformer encoder and test whether they can be used to predict auxiliary labels about tokens or pairs of tokens from external datasets.

Liu et al. (2019) release a suite of probing tasks of varying linguistic complexity, and we largely inherit this setup to study our pruned networks. For streamlined analysis, we broadly split the eighteen tasks into three groups as follows (some tasks may span multiple categories): *Part-of-Speech:* Penn TreeBank POS tagging (**POS**); *Syntactic:* CCG supertagging (**CCG**), parent, grandparent, and greatgrandparent ancestor prediction (**Parent, GParent, GGParent**), chunking (**Chunk**), named entity recognition (**NER**), grammatical error detection (**GED**), conjunct identification (**Conj**), **syntactic arc prediction**, and **syntactic arc classification**; *Semantic:* semantic tagging (**ST**), preposition lexical function and semantic role disambiguation (**PS-Fxn, PS-Role**), event factuality (**EF**), **semantic arc prediction**, **semantic arc classification**, and coreference arc prediction (**Coref**). See Liu et al. (2019) for detailed task descriptions.

Implementation details. We use the same data, splits, and evaluation as Liu et al. (2019). Our initial probing experiments use a single linear layer mapping our 1024-dim token embeddings to the number of task outputs. We train separate probes for each of the six encoder layers for the first nine LTH iterations (LTH0 unpruned, LTH1–8 pruned). See Appendix 1 for further implementation notes.

4.2 Initial results

In Figure 1, we show each LTH model's best probing performance across all layers for each task. Several tasks of varying complexity – POS, CCG, ancestor prediction, syntactic chunking, conjunct ID, syntactic arc classification, semantic tagging, and semantic arc classification – are *sparsity-invariant, i.e.* probing performance does not exhibit
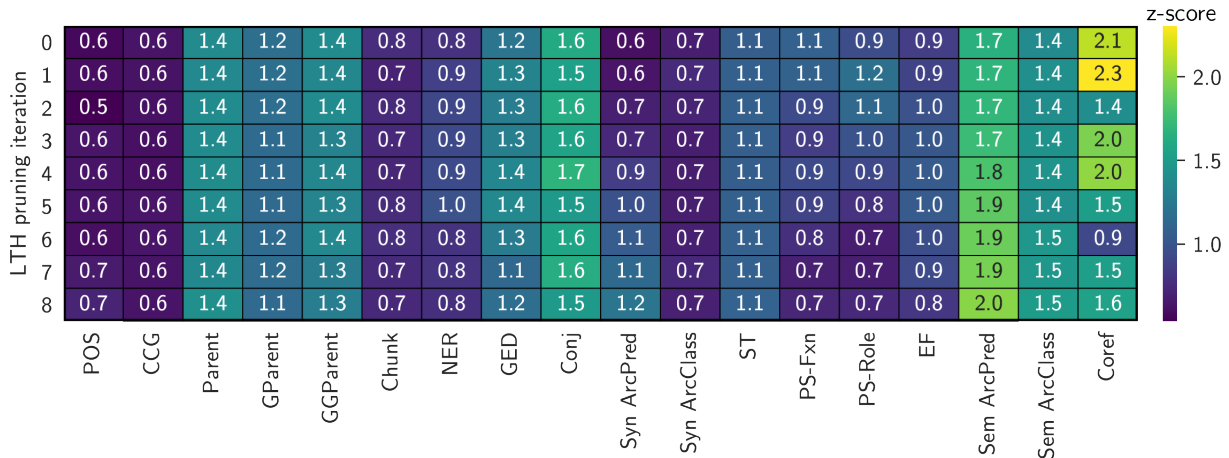
Figure 1: Each cell shows a model's best linear probing performance across all encoder layers for a particular task. Sparsity increases from top to bottom. Values shown are task-specific z-scores.

any sparsity-specific trend. Our metrics for these tasks are on par with probing classifiers trained on *e.g.* BERT contextual word embeddings and are consistently higher than results from a GLoVe baseline (Liu et al., 2019), suggesting that our probes effectively learn these tasks at all sparsities. We conclude that sparsity-invariant tasks encode necessary information for NMT, so their performance will be maintained as long as BLEU remains high enough. Relatedly, we note that some of these tasks (e.g. POS, CCG, ST) have near-100% accuracy and smaller relative improvements over baselines (Liu et al., 2019), so they may only require 'simpler' types of linguistic knowledge contained in any competent language encoder.

Next, there are *sparsity-degrading* tasks, for which the probe does best at an early pruning iteration and starts to drop off at higher sparsities. These tasks include PS-Fxn, PS-Role, and Coref, in which best performance is achieved at LTH0 or LTH1, and performance starts to drop at LTH5 (56.5% sparsity). PS-Fxn classifies a preposition's lexical function in its prepositional phrase, while PS-Role identifies the semantic role that the preposition confers to its object (Schneider et al., 2018). Both tasks require integration of semantic knowledge on top of standard syntax parsing; baseline GLoVe performance is very weak compared to deep contextual representations (Liu et al., 2019). While preposition disambiguation is important to sentence understanding, STREUSLE annotations are likely more fine-grained than necessary for correct translations; even early NMT models could accurately translate most prepositions (Isabelle et al., 2017). As a result, sparse models could plausibly lose some

information relevant to this probing task without impacting BLEU.

Meanwhile, coreference resolution involves identifying pairs of words referring to the same object. NMT models struggle with coreference when semantic information contradicts stereotypical patterns in the training set, *e.g.* for gendered pronouns (Stanovsky et al., 2019). However, these cases are generally rare and investigated with specific challenge sets, and they may not manifest noticeably in test BLEU.

Interestingly, we found that syntactic and semantic arc prediction were *sparsity-improving*; sparser networks consistently performed better. Syntactic arc prediction aims to identify links between co-dependent words in a parse tree, while semantic arc prediction links objects related by the question *Who did What to Whom?* (Oepen et al., 2015). Both tasks are difficult for GLoVe embeddings, but MT-derived representations have done well (Liu et al., 2019; Belinkov et al., 2019). Performance on these arc prediction tasks is lower than on their arc classification counterparts, which were both sparsity-invariant.

Summarizing, we conclude that (1) sparsity-invariant tasks represent core linguistic information that remains encoded as long as BLEU is high enough; (2) there is a push-and-pull with higher-order features as sparsity increases, with some knowledge becoming more readily-available to a probe as other knowledge is degraded.

4.3 Are results probe-sensitive?

Minimal probes offer efficient comparison of language encoders (Hewitt and Liang, 2019), but more

complex probes can also provide useful results when taken in context (Pimentel et al., 2020). We wondered if sparsity-specific performance differences would hold up to a more complex probe, so we repeated a subset of tasks using a two-layer multilayer perceptron. This probe family had over an order of magnitude more parameters than the linear probes. We show performance z-scores in Figure 2, with raw performances available in Table A3.

	PS-Fxn	PS-Role	Coref	SynPred	SemPred
LTH0	1.0	1.1	1.8	2.0	2.0
LTH1	1.0	1.1	1.9	2.0	2.0
LTH2	1.0	1.0	1.5	2.0	2.0
LTH3	0.9	1.0	1.8	2.0	2.0
LTH4	0.8	0.9	1.9	2.0	2.0
LTH5	0.8	0.9	1.5	2.0	2.0
LTH6	0.7	0.8	0.6	2.0	2.0
LTH7	0.8	0.8	1.1	2.0	2.0
LTH8	0.7	0.6	1.1	2.0	2.0

Figure 2: Each model's best performance using the MLP probe, for five tasks whose linear probe performance varied with sparsity. We report z-scores.

Across the board, raw accuracies were higher using the MLP. We no longer see a performance improvement in sparse models for syntactic and semantic arc prediction. Thus, dense model encodings contain the necessary information for these tasks, but a probe must have enough weights to extract it. Given that the Transformer decoder has far more parameters than either probe, it is not surprising that less direct representations of some linguistic features would not impact translation performance. Differences in how directly sparse & dense models encode information is an interesting question perhaps well-suited to recent work on minimum description lengths (Voita and Titov, 2020), but we leave it to future study. Meanwhile, the MLP could not rescue sparse model performance on PS-Fxn, PS-Role, and Coref; results were nearly identical as with the linear probe. We conclude that pruning corrupts some semantic knowledge relevant to these three probing tasks.

4.4 Layer-specific trends

Our analysis so far has focused only on probing performance at the final layer (which always had highest accuracy); we next wanted to study any potential layer-specific sparsity trends. In Figure 3, we show average linear probe performance for each

layer of each model with tasks grouped as syntactic or semantic. For syntactic tasks, performance using layer 1-5 representations increases with sparsity, suggesting that lower layers of sparse models better learn syntactic information. However, performance is maximized and equal across sparsities by layer 6. Results on POS tagging show the same trend (Figure A3). For semantic tasks, all models perform similarly at early layers, while dense models slightly outperform by the final layers. These results support an interpretation in which early sparse layers more directly encode low-level information, whereas dense models tend to rely more on their final layers to (1) equalize differences on syntax tasks and (2) outperform on semantic tasks.

Syntax

Layer	LTH0	LTH1	LTH2	LTH3	LTH4	LTH5	LTH6	LTH7	LTH8
1	-2.1	-2.1	-2.0	-2.0	-1.9	-1.9	-1.9	-1.8	-1.8
2	-0.5	-0.5	-0.4	-0.4	-0.3	-0.3	-0.2	-0.2	-0.2
3	0.1	0.1	0.1	0.2	0.2	0.2	0.2	0.3	0.3
4	0.4	0.5	0.5	0.5	0.5	0.6	0.6	0.7	0.7
5	0.6	0.6	0.6	0.6	0.7	0.8	0.7	0.8	0.8
6	0.8	0.8	0.8	0.8	0.9	0.8	0.9	0.9	0.9

Semantics

Layer	LTH0	LTH1	LTH2	LTH3	LTH4	LTH5	LTH6	LTH7	LTH8
1	-1.6	-1.5	-1.5	-1.4	-1.2	-1.4	-1.6	-1.4	-1.3
2	-0.2	-0.2	-0.3	-0.2	-0.2	-0.3	-0.4	-0.3	-0.2
3	0.1	0.1	-0.0	0.0	-0.1	-0.1	-0.3	-0.3	-0.3
4	0.2	0.4	0.3	0.4	0.3	0.3	0.2	0.3	0.2
5	0.3	0.4	0.5	0.4	0.4	0.4	0.3	0.2	0.3
6	1.2	1.2	1.1	1.2	1.2	1.1	1.0	1.1	1.1

Figure 3: Each cell shows the average probing z-score (across all syntactic or semantic tasks) for a particular layer of a particular model. Model sparsity increases from left to right.

5 Measuring Behavioral Similarities of Sparse & Dense Models

5.1 Experimental setup

Probing classifiers are one method of studying linguistic knowledge, but even two models with similar probing results may have divergent representations (Saphra and Lopez, 2019). We are left with the question: do sparse models learn to arrive at the same internal representations as dense models using fewer weights (up to linear transformation), or do their activations and attention maps shift alto-

gether? To answer this question, we perform direct, unsupervised study of our model's internal vectors on unseen text.

First, we deploy our unpruned model and the eight pruned models on our 3000 validation sentences, which span 120K tokens total. We store the 1024-dim token representations and the 16 heads of each of the three attention types (enc-enc, enc-dec, dec-dec) at each layer. To compute similarities from these data, we adopt a subset of the metrics described in Wu et al. (2020). Each metric offers a distinct lens of viewing the behavioral similarity of a layer pair (L, L').

NeuronSim (Bau et al., 2019) is a local similarity measure that quantifies how well the individual neurons k of a layer L (*i.e.*, a single dimension of a layer's representation) align with individual neurons k' in another layer L':

$$\text{NeuronSim}(L, L') = \underset{k \in L}{\text{mean}} \left\{ \max_{k' \in L'} \text{corr}(k, k') \right\},$$

where $\text{corr}(k, k')$ is the mean Pearson correlation between the activations of neurons k and k' across all tokens. **LayerSim** is a global similarity measure that quantifies how well entire layer representations align. For two layers, LayerSim compares their vectors of 1024-dim representations across all tokens. We use the linear centered kernel alignment (linearCKA) (Kornblith et al., 2019) as our similarity metric. Analogous to LayerSim, **AttentionSim** compares the attention distributions across all heads between two different layers. For every sentence s in our dataset, we extract the 16 head attention vector $\boldsymbol{\alpha}_{ij}(s)$ for each word pair $(w_i, w_j) \in s$. We then apply linearCKA, treating each pair of words as a 16-dimensional example.

5.2 Similarity of activations

Looking first at NeuronSim, we find that neuron function in the encoder and decoder is largely conserved as sparsity increases. Across all neurons k in layer L of LTH0 (unpruned), k's most correlated neuron in layer L of LTH8 (70% sparse) is the same neuron k, 99.8% of the time. However, the *magnitude* of neuron similarity with the unpruned network consistently drops with sparsity (Figure 4). In both the encoder and decoder, this drop was sharpest for higher layers: *e.g.* between LTH0 and LTH8, decoder layer 1 had 0.82 NeuronSim while decoder layer 6 had 0.71 NeuronSim.

Next, we wondered if all neurons gradually become less similar to their unpruned selves, or rather
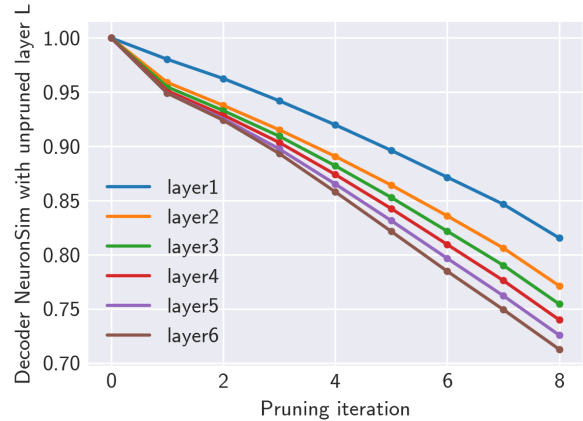

Figure 4: Each decoder layer's NeuronSim with the corresponding LTH0 layer; sparsity increases left to right.

if some neurons remain the same whereas others "drop out" or change functions entirely. Visualizing the distributions of neuron correlations revealed the former: all sparse-dense neuron pairs became less similar during pruning (Figure A4). We therefore conclude that as sparsity increases, (1) neurons gradually diverge from their dense counterparts, and (2) neurons in higher layers diverge more rapidly than neurons in lower layers.

In Figure 5, we compute encoder LayerSim scores between LTH0 (dense) and LTH8 (sparse). Like NeuronSim, we find that similarity decreases with sparsity, especially at higher layers. Interestingly, in the decoder, LayerSim between dense and sparse was consistently higher at layer 6 than layer 5 (Figure A5), perhaps because layer 6 representations 'converge' before final token prediction. Comparing off-diagonal layer similarities between encoder and decoder, we find that different decoder layers are less similar than different encoder layers (Figures 5, A6). That is, each decoder layer changes token representations more significantly than each encoder layer, suggesting that decoder layers perform more processing than encoder layers (perhaps due to the additional parameterization afforded by the encoder-decoder attention module).

Next, we find that early sparse model representations are closer to their final representations than early dense representations are. In the encoder, for example, sim(dense-2, dense-6) is 0.67, while sim(sparse-2, sparse-6) is 0.74 (Figure 5). In the decoder, sim(dense-4, dense-6) is 0.80 while sim(sparse-4, sparse-6) is 0.92. In general, early and late layers are more similar in sparse models, and this trend strengthens as sparsity increases.

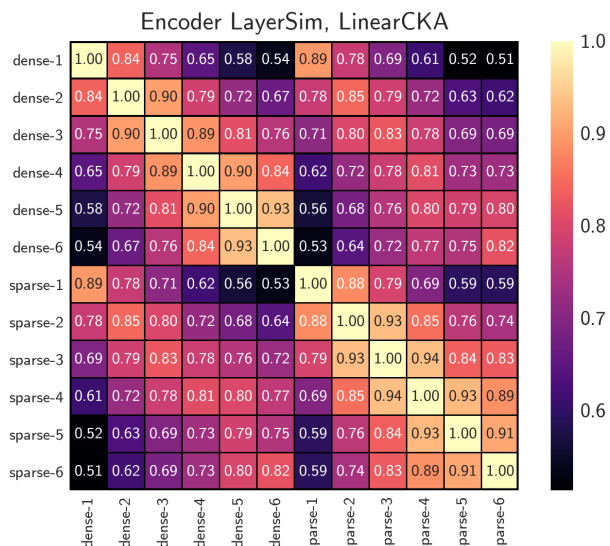We hypothesized two explanations: (1) Sparse

Figure 5: Encoder layer similarities for pairs of layers in LTH0 (dense) and LTH8 (70% sparse).

models have less complex final representations, so early layers are inherently not as 'far away.' (2) Early sparse layers learn more effective representations, so they are closer to the final ones. Looking at decoder LayerSim values, we found that early sparse representations were often more similar to final *dense* representations than early dense representations were, providing some evidence for (2). For example, sim(sparse-2, dense-6) is 0.55 while sim(dense-2, dense-6) is 0.49 (Figure A6).

5.3 Analyzing the final layers

Late encoder and decoder representations exhibited largest difference between sparse and dense models. To further characterize the difference, we computed the SVD of each model's final encoder and decoder layer representation matrices. We find that sparse models have more variance explained by the top k singular vectors (Figure A7). For example, for the encoder, 80% variance requires $k = 290$ for LTH0 but just $k = 176$ for LTH8; for the decoder, $k = 283$ and $k = 139$. We conclude that final layer representations in sparse models fundamentally have less mathematical complexity (which is not necessarily an obvious result for magnitude pruning, versus *e.g.* in the case of pruning entire neurons).

We next found word categories for which dense and sparse encoder representations differed most. We used linearCKA to compute similarities of tokens grouped by frequency bin, POS (Penn Tree-Bank), or semantic tag (Bjerva et al., 2016).

Mid-frequency tokens (rank 100 to 500) had highest similarity (0.95), while both common (rank 0 to 5) and rare (rank 2500 or higher) tokens had similarity 0.87. For POS, coordinating conjunctions and superlative adjectives had highest similarity (0.98 and 0.97), while proper nouns and particles were lowest (0.85, 0.86). Models also learned very different representations of punctuation, *e.g.* with the possessive ending and period tokens at 0.79 and 0.84 similarity respectively. For semantic tags, the least similar classes (0.84) were perfect/progressive verb tense modifiers, *e.g.* 'has arrived', 'is running', etc. The broad 'concept' class spanning uncommon nouns also had relatively dissimilar (0.87) representations. Together, these results suggest that sparse model encodings differ most for (1) tokens with several syntactic/semantic meanings to disambiguate and (2) rare words.

5.4 Attention-level similarities

5.4.1 Encoder self-attention

Unlike the encoder activations, encoder self-attention distributions remain remarkably similar between sparse and dense models. For layers 2–6, AttentionSim scores on the sparse-dense diagonal are very close to 1 (Figure 6, bottom left). Layer 1 is an anomaly, with sparse and dense attentions differing widely (0.62). The first layer's attention distributions in the sparse model become much more similar to its later layers (average 0.41) than in the dense model (average 0.18), suggesting that this first self-attention layer learns more salient relationships in the sparse model.

5.4.2 Encoder-decoder attention

Compared to self-attention, encoder-decoder attention displays more variation across layers and sparsities (Figure A8). While many off-diagonal similarities exceed 0.85 in self-attention, off-diagonal encoder-decoder similarities are often less than 0.7. In particular, the first three enc-dec layers differ strongly from the last three layers, demonstrating heterogeneity in learned attention distributions at different levels of decoding.

As sparsity increases, attention maps at a given layer remain mostly consistent with the dense model, although there is slightly more deviation than in self-attention. Interestingly, the model's off-diagonal similarities gradually increase with sparsity (average 0.69 in LTH8 vs. 0.63 in LTH0), particularly with sparse-5 becoming more similar to sparse-1,2,3. Sparsity may have a dampening
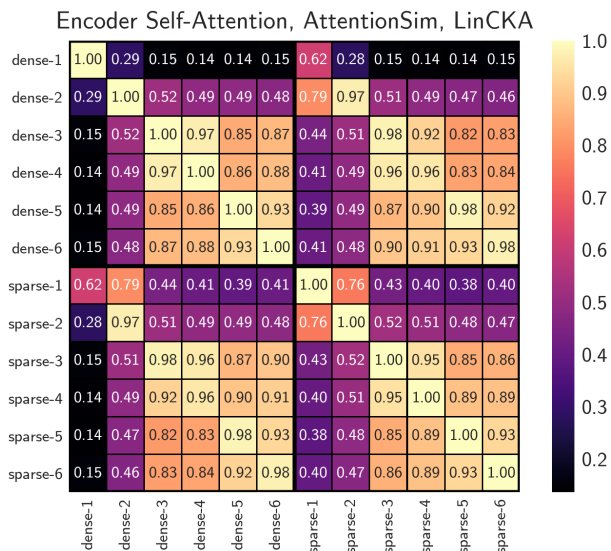
Figure 6: Encoder self-attention similarities for pairs of layers in LTH0 (dense) and LTH8 (70% sparse).

effect on the distinctions of individual encoder-decoder layers; these more homogenous attention distributions may explain the drop in the decoder's total representational complexity (Figure A7).

5.4.3 Decoder self-attention

Decoder self-attention distributions in pruned models are almost identical (0.99 similarity) to their corresponding unpruned distributions, even more so than enc-self and enc-dec attention (Figure A8). We attribute this phenomenon to the relative simplicity of the decoder self-attention module, and in consequence the relative ease at which weights can be pruned without changing expressivity. For instance, we found that 40% of all decoder self-attention distributions in the sparse model (41% in the dense model) placed over 0.95 of the probability mass on a single query token, compared to only 30% and 16% of the distributions in enc-self and enc-dec attention respectively.

Further, all similarity scores between different layers (off-diagonal) in dec self-attention are significantly higher than in encoder self and enc-dec attention. That is, decoder self-attention is homogenous across layers. Despite the simple nature of this self-attention, the decoder can still learn complex representations due to its pairing with the powerful encoder-decoder attention module.

6 Discussion

A consistent theme in our analysis is the behavioral shift of early layers (1–3), which occurs gradu-

ally as sparsity increases. Our probing results find that lower layers of sparse models more directly encode POS and syntax information compared to dense models, even though performance of the final encoder representations is similar (4.4). Moreover, our similarity analyses conclude that early layer encoder hidden representations (5.2) and attention distributions (5.4.1) trend closer towards their respective final representations in sparse models. Information-theoretically, sparse layers have less maximum capacity for encoding, so each individual layer must shoulder more load for the final representations to remain predictively salient. Conversely, an overparameterized dense model can compensate for weak lower layer representations with its upper layers. Indeed, upper FC layers are pruned more than lower FC layers (3.4), reflecting the shift in modeling power away from higher layers.

We also observe a gradual loss of information stored in model representations as weights are pruned, especially in later layers. Individual neurons diverge from their dense counterparts (5.2), causing a drop in overall representational complexity in the encoder and decoder. Correspondingly, sparse models perform worse at higher-order semantic tasks that are less relevant to BLEU (4.3). The reduced overall complexity of sparse representations may partially explain why final layers are observed to be closer to early layers (5.2, 5.4.2).

Finally, we find that sparse models' attention distributions remain largely similar to their values in the dense model. This ability to reduce weights in attention modules while maintaining nearly identical representations affirms other lines of work (Guo et al., 2019; Wang et al., 2020). Of the three attention types, encoder-decoder is pruned least (3.4), varies most across sparsities, and exhibits most within-model, inter-layer heterogeneity (5.4.3). These results corroborate existing evidence of its unique importance (Voita et al., 2019; Michel et al., 2019). Meanwhile, decoder self-attention is extremely homogenous across layers and sparsities, perhaps because encoder-decoder attention is more relevant to creating rich representations.

Limitations. Our work focuses on pruned Transformers for which BLEU remains similar to the original model. However, BLEU is an imperfect measure of translation quality (Callison-Burch et al., 2006), and it is possible that our pruned models actually perform worse on the task at lower

sparsities than suggested by BLEU. Still, we think our work is relevant given that sparse models are typically only held to the standard of matching unpruned task performance.

Next, we emphasize that our work focuses solely on magnitude pruning, which may not be representative of how other pruning methods impact Transformers. We chose this style of pruning primarily because it allows for higher overall sparsity without drop in performance (Renda et al., 2020). Further, while it might be expected (and has been shown, in some cases) that pruning entire neurons or attention heads would substantially change *e.g.* the distributions of the model's outputs, we found less existing work specifically measuring the effects of magnitude pruning. This dearth of analysis seemed particularly egregious given recent growth in work on unstructured sparsity (Blalock et al., 2020).

Finally, a note on probing classifiers: as has been widely discussed by the community (*e.g.* Pimentel et al. (2020)), probes measure *correlation* between model outputs and auxilliary information. Differences in probe performance do not necessarily imply anything about what information actually uses during its forward pass. Especially since we find some evidence suggesting that sparse models may be encoding information across layers, it is possible that their differing structure may explain worse probe performance, as opposed to fundamentally weaker linguistic feature extraction. We hope future work supplements our results by analyzing a model's encoded knowledge in other ways.

7 Conclusions

We evaluate how unstructured pruning affects the behavior of Transformers while task performance is maintained. We use probing classifiers to demonstrate that pruning degrades semantic knowledge before affecting BLEU, and that early layers of sparse models better encode low-level linguistic information. Unsupervised similarity analysis reveals that pruning induces representational changes in the encoder and decoder, particularly in higher layers, and that early sparse representations are more similar to their final representations. Meanwhile, attention distributions remain remarkably similar, even at high sparsities.

Acknowledgements

References

Anthony Bau, Yonatan Belinkov, Hassan Sajjad, Nadir Durrani, Fahim Dalvi, and James Glass. 2019. Identifying and controlling important neurons in neural machine translation.

Yonatan Belinkov, Nadir Durrani, Fahim Dalvi, Hassan Sajjad, and James Glass. 2017. What do neural machine translation models learn about morphology? *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Yonatan Belinkov, Nadir Durrani, Fahim Dalvi, Hassan Sajjad, and James Glass. 2019. On the Linguistic Representational Power of Neural Machine Translation Models. *arXiv:1911.00317 [cs]*. ArXiv: 1911.00317.

Yonatan Belinkov and James Glass. 2018. Analysis methods in neural language processing: A survey.

Johannes Bjerva, Barbara Plank, and Johan Bos. 2016. Semantic Tagging with Deep Residual Networks. *arXiv:1609.07053 [cs]*. ArXiv: 1609.07053.

Davis Blalock, Jose Javier Gonzalez Ortiz, Jonathan Frankle, and John Guttag. 2020. What is the State of Neural Network Pruning? *arXiv:2003.03033 [cs, stat]*. ArXiv: 2003.03033.

Christopher Brix, Parnia Bahar, and Hermann Ney. 2020. Successfully applying the stabilized lottery ticket hypothesis to the transformer architecture. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3909–3915, Online. Association for Computational Linguistics.

Chris Callison-Burch, Miles Osborne, and Philipp Koehn. 2006. Re-evaluating the role of Bleu in machine translation research. In *11th Conference of the European Chapter of the Association for Computational Linguistics*, Trento, Italy. Association for Computational Linguistics.

Tianlong Chen, Jonathan Frankle, Shiyu Chang, Sijia Liu, Yang Zhang, Zhangyang Wang, and Michael Carbin. 2020. The lottery ticket hypothesis for pretrained bert networks.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. 2019. What does bert look at? an analysis of bert's attention. *ArXiv*, abs/1906.04341.

Alexis Conneau, German Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. 2018. What you can cram into a single vector: Probing sentence embeddings for linguistic properties.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding.

Angela Fan, Edouard Grave, and Armand Joulin. 2019. Reducing Transformer Depth on Demand with Structured Dropout. *arXiv:1909.11556 [cs, stat]*. ArXiv: 1909.11556.

Jonathan Frankle, Gintare Karolina Dziugaite, Daniel M. Roy, and Michael Carbin. 2019. Stabilizing the lottery ticket hypothesis.

Prakhar Ganesh, Yao Chen, Xin Lou, Mohammad Ali Khan, Yin Yang, Deming Chen, Marianne Winslett, Hassan Sajjad, and Preslav Nakov. 2020. Compressing Large-Scale Transformer-Based Models: A Case Study on BERT. *arXiv:2002.11985 [cs, stat]*. ArXiv: 2002.11985.

Mitchell A. Gordon, Kevin Duh, and Nicholas Andrews. 2020. Compressing bert: Studying the effects of weight pruning on transfer learning.

Qipeng Guo, Xipeng Qiu, Pengfei Liu, Yunfan Shao, Xiangyang Xue, and Zheng Zhang. 2019. Star-transformer.

Yaru Hao, Li Dong, Furu Wei, and Ke Xu. 2019. Visualizing and Understanding the Effectiveness of BERT. *arXiv:1908.05620 [cs]*. ArXiv: 1908.05620.

John Hewitt and Percy Liang. 2019. Designing and Interpreting Probes with Control Tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2733–2743, Hong Kong, China. Association for Computational Linguistics.

Pierre Isabelle, Colin Cherry, and George Foster. 2017. A Challenge Set Approach to Evaluating Machine Translation. *arXiv:1704.07431 [cs]*. ArXiv: 1704.07431.

Taeuk Kim, Jihun Choi, Daniel Edmiston, and Sang goo Lee. 2020. Are pre-trained language models aware of phrases? simple but strong baselines for grammar induction.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. 2019. Similarity of Neural Network Representations Revisited. *arXiv:1905.00414 [cs, q-bio, stat]*. ArXiv: 1905.00414.

Nelson F. Liu, Matt Gardner, Yonatan Belinkov, Matthew E. Peters, and Noah A. Smith. 2019. Linguistic Knowledge and Transferability of Contextual Representations. *arXiv:1903.08855 [cs]*. ArXiv: 1903.08855.

Paul Michel, Omer Levy, and Graham Neubig. 2019. Are Sixteen Heads Really Better than One? *arXiv:1905.10650 [cs]*. ArXiv: 1905.10650.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Silvie Cinková, Dan Flickinger, Jan Hajič, and Zdeňka Urešová. 2015. SemEval 2015 Task 18: Broad-Coverage Semantic Dependency Parsing. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 915–926, Denver, Colorado. Association for Computational Linguistics.

Tiago Pimentel, Josef Valvoda, Rowan Hall Maudslay, Ran Zmigrod, Adina Williams, and Ryan Cotterell. 2020. Information-Theoretic Probing for Linguistic Structure. *arXiv:2004.03061 [cs]*. ArXiv: 2004.03061.

Sai Prasanna, Anna Rogers, and Anna Rumshisky. 2020. When bert plays the lottery, all tickets are winning.

Alessandro Raganato and Jörg Tiedemann. 2018. An analysis of encoder representations in transformer-based machine translation. In *BlackboxNLP@EMNLP*.

Alex Renda, Jonathan Frankle, and Michael Carbin. 2020. Comparing Rewinding and Fine-tuning in Neural Network Pruning. *arXiv:2003.02389 [cs, stat]*. ArXiv: 2003.02389.

Jonathan S. Rosenfeld, Jonathan Frankle, Michael Carbin, and Nir Shavit. 2020. On the Predictability of Pruning Across Scales. *arXiv:2006.10621 [cs, stat]*. ArXiv: 2006.10621.

Hassan Sajjad, Fahim Dalvi, Nadir Durrani, and Preslav Nakov. 2020. Poor man's bert: Smaller and faster transformer models.

Naomi Saphra and Adam Lopez. 2019. Understanding Learning Dynamics Of Language Models with SVCCA. *Proceedings of the 2019 Conference of the North*, pages 3257–3267. ArXiv: 1811.00225.

Nathan Schneider, Jena D. Hwang, Vivek Srikumar, Jakob Prange, Austin Blodgett, Sarah R. Moeller, Aviram Stern, Adi Bitan, and Omri Abend. 2018. Comprehensive Supersense Disambiguation of English Prepositions and Possessives. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 185–196, Melbourne, Australia. Association for Computational Linguistics.

Xing Shi, Inkit Padhi, and Kevin Knight. 2016. Does string-based neural MT learn source syntax? In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1526–1534, Austin, Texas. Association for Computational Linguistics.

Gabriel Stanovsky, Noah A. Smith, and Luke Zettle-moyer. 2019. Evaluating Gender Bias in Machine Translation. *arXiv:1906.00591 [cs]*. ArXiv: 1906.00591.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need.

Elena Voita, Pavel Serdyukov, Rico Sennrich, and Ivan Titov. 2018. Context-aware neural machine translation learns anaphora resolution. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1264–1274, Melbourne, Australia. Association for Computational Linguistics.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. 2019. Analyzing Multi-Head Self-Attention: Specialized Heads Do the Heavy Lifting, the Rest Can Be Pruned. *arXiv:1905.09418 [cs]*. ArXiv: 1905.09418.

Elena Voita and Ivan Titov. 2020. Information-Theoretic Probing with Minimum Description Length. *arXiv:2003.12298 [cs.CL]*.

Sinong Wang, Belinda Z. Li, Madian Khabsa, Han Fang, and Hao Ma. 2020. Linformer: Self-attention with linear complexity.

John Wu, Yonatan Belinkov, Hassan Sajjad, Nadir Durrani, Fahim Dalvi, and James Glass. 2020. Similarity Analysis of Contextual Word Representation Models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4638–4655, Online. Association for Computational Linguistics.

Haonan Yu, Sergey Edunov, Yuandong Tian, and Ari S. Morcos. 2020. Playing the lottery with rewards and multiple languages: lottery tickets in RL and NLP. *arXiv:1906.02768 [cs, stat]*. ArXiv: 1906.02768.